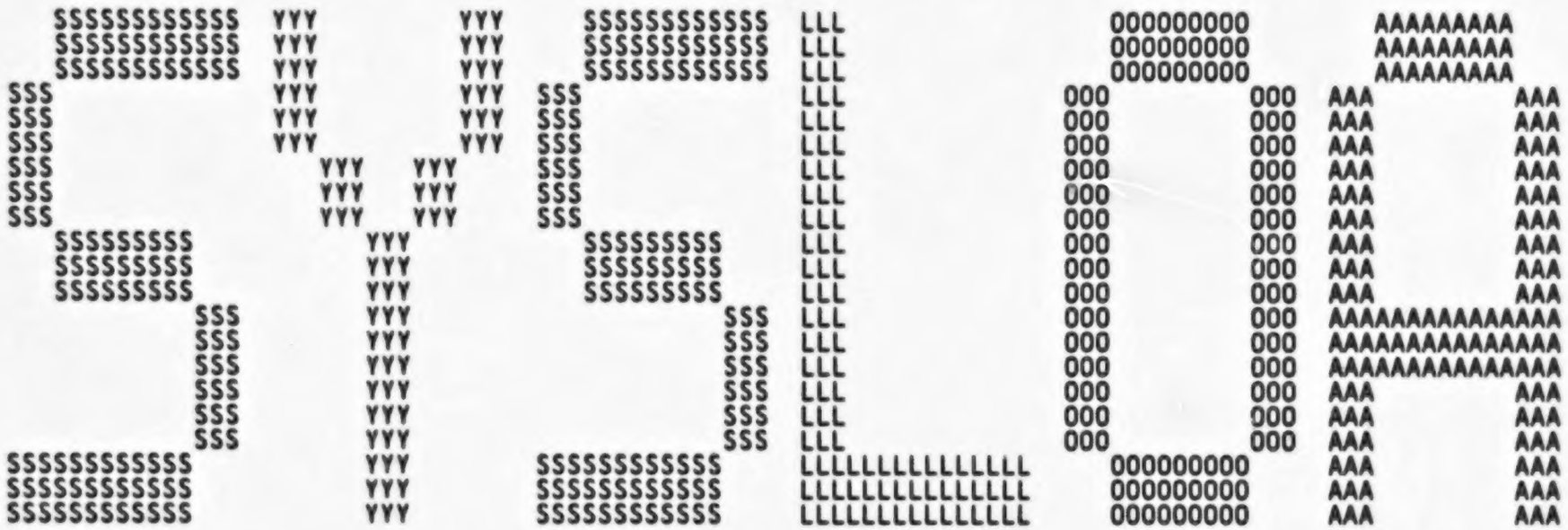
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL OOOOOOOOO AAAAAAAAA
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL OOOOOOOOO AAAAAAAAA
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL OOOOOOOOO AAAAAAAAA
SSS YYY YYY SSS LLL OOO OOO AAA AAA
SSS YYY YYY SSS LLL OOO OOO AAA AAA
SSS YYY YYY SSS LLL OOO OOO AAA AAA
SSS YYY YYY SSS LLL OOO OOO AAA AAA
SSS YYY YYY SSS LLL OOO OOO AAA AAA
SSS YYY YYY SSS LLL OOO OOO AAA AAA
SSSSSSSSS YYY SSSSSSSSS LLL OOO OOO AAA AAA
SSSSSSSSS YYY SSSSSSSSS LLL OOO OOO AAA AAA
SSSSSSSSS YYY SSSSSSSSS LLL OOO OOO AAA AAA
SSS YYY SSS LLL OOO OOO AAAAAAAAAAAAAAA
SSS YYY SSS LLL OOO OOO AAAAAAAAAAAAAAA
SSS YYY SSS LLL OOO OOO AAAAAAAAAAAAAAA
SSS YYY SSS LLL OOO OOO AAA AAA
SSS YYY SSS LLL OOO OOO AAA AAA
SSSSSSSSSSS YYY SSSSSSSSSSS LLLLLLLLLLLLLLL OOOOOOOOO AAA AAA
SSSSSSSSSSS YYY SSSSSSSSSSS LLLLLLLLLLLLLLL OOOOOOOOO AAA AAA
SSSSSSSSSSS YYY SSSSSSSSSSS LLLLLLLLLLLLLLL OOOOOOOOO AAA AAA

```
DDDDDDDD      SSSSSSSS  TTTTTTTTTT  RRRRRRRR    DDDDDDDD    LL           CCCCCCCC  KK      KK
DDDDDDDD      SSSSSSSS  TTTTTTTTTT  RRRRRRRR    DDDDDDDD    LL           CCCCCCCC  KK      KK
DD      DD  SS              TT      RR      RR  DD      DD  LL         CC          KK      KK
DD      DD  SS              TT      RR      RR  DD      DD  LL         CC          KK      KK
DD      DD  SS              TT      RR      RR  DD      DD  LL         CC          KK    KK
DD      DD  SS              TT      RR      RR  DD      DD  LL         CC          KK    KK
DD      DD    SSSSSS        TT      RRRRRRRR    DD      DD  LL         CC          KKKKKK
DD      DD    SSSSSS        TT      RRRRRRRR    DD      DD  LL         CC          KKKKKK
DD      DD          SS      TT      RR  RR      DD      DD  LL         CC          KK    KK
DD      DD          SS      TT      RR  RR      DD      DD  LL         CC          KK    KK
DD      DD          SS      TT      RR    RR    DD      DD  LL         CC          KK      KK      ....
DD      DD          SS      TT      RR    RR    DD      DD  LL         CC          KK      KK      ....
DDDDDDDD    SSSSSSSS        TT      RR      RR  DDDDDDDD    LLLLLLLLLL   CCCCCCCC  KK      KK      ....
DDDDDDDD    SSSSSSSS        TT      RR      RR  DDDDDDDD    LLLLLLLLLL   CCCCCCCC  KK      KK      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

DST
V04

Table of contents

(2) 70 DECLARATIONS
(3) 157 LCK$SND_TIMESTAMP_RQST
(4) 286 GET_TIMESTAMP - Get a bitmap time stamp
(5) 356 LCK$RCV_TIMESTAMP_RQST
(6) 437 CHECK_TIMESTAMP - Check bitmap timestamp
(7) 503 LCK$SND_SRCHDLCK - Send deadlock search message
(8) 728 LCK$RCV_SRCHDLCK - Receive search deadlock message
(9) 846 LCK$SND_DLCKFND - Send deadlock found message
(10) 943 LCK$RCV_DLCKFND - Receive deadlock found message
(11) 981 LCK$SND_REDO_SRCH - Send a redo deadlock search message
(12) 1046 LCK$RCV_REDO_SRCH
(13) 1077 REDO_SRCH - Redo deadlock search
(14) 1163 LCK$CVT_ID_TO_LKB - Convert a lockid to LKB address
(15) 1206 SEND_DLCK_MSG - Send any deadlock detection message
(16) 1243 RCV_DLCK_MSG - Receive a deadlock message
(17) 1287 DEALL_DLCK_MSG - Deallocate deadlock message buffer
(18) 1321 LCK$ALLOC_LONGCDRP - Allocate a long CDRP
(19) 1365 WAIT_FOR_POOL - Wait for pool

```
0000     1          .TITLE  DSTRDLCK - DISTRIBUTED DEADLOCK DETECTION AND RESOLUTION
0000     2          .IDENT  'V04-000'
0000     3
0000     4  ;****************************************************************************
0000     5  ;*                                                                          *
0000     6  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000     7  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8  ;*  ALL RIGHTS RESERVED.                                                    *
0000     9  ;*                                                                          *
0000    10  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    11  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    12  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    13  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    14  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    15  ;*  TRANSFERRED.                                                            *
0000    16  ;*                                                                          *
0000    17  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    18  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    19  ;*  CORPORATION.                                                            *
0000    20  ;*                                                                          *
0000    21  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    22  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    23  ;*                                                                          *
0000    24  ;*                                                                          *
0000    25  ;****************************************************************************
0000    26
0000    27  ;++
0000    28  ; FACILITY: EXECUTIVE, SYSTEM SERVICES
0000    29  ;
0000    30  ; ABSTRACT:
0000    31  ;       This module implements distributed deadlock detection (and resolution)
0000    32  ;       for the VMS lock manager system services ($ENQ and $DEQ) when
0000    33  ;       operating in a VAXcluster environment.
0000    34  ;
0000    35  ; ENVIRONMENT: VAX/VMS, VAXcluster loadable code
0000    36  ;
0000    37  ; AUTHOR: Steve Beckhardt,       CREATION DATE: 28-Feb-1984
0000    38  ;
0000    39  ; MODIFIED BY:
0000    40  ;
0000    41  ;       V03-006 SRB0143         Steve Beckhardt                 9-Jul-1984
0000    42  ;               Changed handling of repeated failures to complete a deadlock
0000    43  ;               search.  Instead of calling it a deadlock, the lock is now
0000    44  ;               placed back on the end of the time out queue with a fresh
0000    45  ;               wait time and retry count.  This should eliminate the
0000    46  ;               occasional false deadlocks.  As a result, the maximum
0000    47  ;               timestamp lifetime was reduced to 5 (1.6 secs.)
0000    48  ;
0000    49  ;       V03-005 SRB0137         Steve Beckhardt                 9-Jul-1984
0000    50  ;               Fixed bugs in timestamp lifetime code.  Increased
0000    51  ;               maximum timestamp lifetime to 6 (3.2 secs.)
0000    52  ;
0000    53  ;       V03-004 SRB0134         Steve Beckhardt                 22-Jun-1984
0000    54  ;               Fixed bug in stack handling in insufficient pool code.
0000    55  ;
0000    56  ;       V03-003 SRB0130         Steve Beckhardt                 18-May-1984
0000    57  ;               Fixed bug involving location of test for NODLCKWT flag.
```

```
0000     58 ;
0000     59 ;       V03-002 SRB0125         Steve Beckhardt                 2-May-1984
0000     60 ;               Fixed bug involving race between process level handling
0000     61 ;               of lock granted message (getting lock on PCB queue) and
0000     62 ;               receiving deadlock search message.
0000     63 ;
0000     64 ;       V03-001 SRB0119         Steve Beckhardt                 10-Mar-1984
0000     65 ;               Fixed bugs, added support for LCK$M_NODLCKWT flag.
0000     66 ;               Added support for waiting for pool.
0000     67 ;
0000     68 ;--
```

```
                     0000     70          .SBTTL  DECLARATIONS
                     0000     71 ;
                     0000     72 ; INCLUDE FILES:
                     0000     73 ;
                     0000     74
                     0000     75 ;
                     0000     76 ; EXTERNAL SYMBOLS:
                     0000     77 ;
                     0000     78
                     0000     79          $CADEF                          ; Conditional assembly switches
                     0000     80          $CDRPDEF                        ; CDRP offsets
                     0000     81          $CLSMSGDEF                      ; Cluster message offsets
                     0000     82          $CLUBDEF                        ; CLUB offsets
                     0000     83          $CSBDEF                         ; CSB offsets
                     0000     84          $DYNDEF                         ; Structure type code definitions
                     0000     85          $FKBDEF                         ; Fork block offsets
                     0000     86          $IPLDEF                         ; IPL definitions
                     0000     87          $LCKDEF                         ; LCK definitions
                     0000     88          $LKBDEF                         ; LKB offsets
                     0000     89          $PCBDEF                         ; PCB offsets
                     0000     90          $RSBDEF                         ; RSB offsets
                     0000     91          $SSDEF                          ; System status code definitions
                     0000     92
                     0000     93 ;
                     0000     94 ; MACROS:
                     0000     95 ;
                     0000     96
                     0000     97 ;
                     0000     98 ; EQUATED SYMBOLS:
                     0000     99 ;
                     0000    100
0007A120             0000    101 TSLT_UNITS = 50*1000*10                 ; Time stamp lifetime units (50 ms.)
00000005             0000    102 MAX_TSLT = 5                            ; Maximum timestamp lifetime shift count
                     0000    103                                         ; (represents 1.6 secs.)
                     0000    104                                         ; This also represents the interval
                     0000    105                                         ; that must elapse before local searches
                     0000    106                                         ; can proceed without timestamps
                     0000    107
00000018             0000    108 LOCKFRAME = 24                          ; Number of bytes pushed onto
                     0000    109                                         ; stack for each recursive call
                     0000    110                                         ; of SEARCH_RESDLCK  (5 registers
                     0000    111                                         ; plus return address).  This
                     0000    112                                         ; cannot be changes without making
                     0000    113                                         ; corresponding coding changes
                     0000    114                                         ; This must also agree with the
                     0000    115                                         ; symbol of the same name in
                     0000    116                                         ; DEADLOCK.MAR.
                     0000    117
                     0000    118 ;
                     0000    119 ; OWN STORAGE:
                     0000    120 ;
                     0000    121
                00000000     122          .PSECT  $$$040,LONG
                     0000    123
                     0000    124          .ALIGN  LONG
                     0000    125
                     0000    126 ;
```

```
                     0000   127 ; NOTE:  The fork block and lock message buffer must be adjacent
                     0000   128 ;
                     0000   129 LKMSG_FKB:
00000000 00000000    0000   130         .QUAD   0                       ; Queue links
              0018   0008   131         .WORD   FKB$K_LENGTH            ; Size
                08   000A   132         .BYTE   DYN$C_FRK               ; Type
                08   000B   133         .BYTE   IPL$_SYNCH              ; Fork IPL
          00000018   000C   134         .BLKB   FKB$K_LENGTH-12         ; Remainder of fork block
                     0018   135 LKMSG_BFR:                              ; Buffer to use for lock messages
          0000004C   0018   136         .BLKB   LKMSG$K_DLM_LENGTH
                     004C   137
                     004C   138 ;****************************************************************************
                     004C   139 ;
                     004C   140 ; NOTE:  The following assumptions are in effect for this entire module
                     004C   141 ;
                     004C   142 ;****************************************************************************
                     004C   143
                     004C   144         ASSUME  LKMSG$B_TSLT            EQ      2+LKMSG$W_MEMSEQ
                     004C   145         ASSUME  LKMSG$L_ORIGEPID        EQ      2+LKMSG$B_TSLT
                     004C   146         ASSUME  LKMSG$L_ORIGLKID        EQ      4+LKMSG$L_ORIGEPID
                     004C   147         ASSUME  LKMSG$L_ORIGCSID        EQ      4+LKMSG$L_ORIGLKID
                     004C   148         ASSUME  LKMSG$Q_BITMAP_EXP      EQ      4+LKMSG$L_ORIGCSID
                     004C   149         ASSUME  LKMSG$L_VCTMPRT         EQ      8+LKMSG$Q_BITMAP_EXP
                     004C   150         ASSUME  LKMSG$L_VCTMLKID        EQ      4+LKMSG$L_VCTMPRT
                     004C   151         ASSUME  LKMSG$L_VCTMCSID        EQ      4+LKMSG$L_VCTMLKID
                     004C   152         ASSUME  LKMSG$L_NEXTLKID        EQ      4+LKMSG$L_VCTMCSID
                     004C   153
                     004C   154
                00000000    155         .PSECT  $$$020
```

```
                                        .SBTTL  LCK$SND_TIMESTAMP_RQST

                                ;++
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ;       This routine sends a timestamp request to the system issuing
                                ;       timestamps unless this system is issuing timestamps.  If a message
                                ;       is actually sent, then this routine does not return to it's caller.
                                ;       Instead, the stack is unwound and we exit deadlock detection.
                                ;       This routine only returns to its caller if this system
                                ;       is issuing timestamps and we successfully get one.
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       BSBW    LCK$SND_TIMESTAMP_RQST
                                ;       Note:   This routine only returns to its caller if a timestamp
                                ;               is issued locally.  In all other cases, the stack is
                                ;               reset and we exit from deadlock detection.
                                ;
                                ; INPUT PARAMETERS:
                                ;
                                ;       R8      EPID of original process
                                ;       R10     Stack position to unwind to
                                ;
                                ; IMPLICIT INPUTS:
                                ;
                                ;       It is assumed that the lock that started the deadlock search is
                                ;       still at the head of the timeout queue.
                                ;
                                ; OUTPUT PARAMETERS:
                                ;
                                ;       R9      Address of a message buffer template to be used instead
                                ;               of a real message buffer if a timestamp is assigned locally
                                ;
                                ; SIDE EFFECTS:
                                ;
                                ;       The bitmap is cleared if we issue a time stamp
                                ;--

                                LCK$SND_TIMESTAMP_RQST::
54  00000000'GF    D0           MOVL    G^LCK$GL_TIMOUTQ,R4     ; Get original lock (from head of queue)
53  00000000'EF    D0           MOVL    LCK$GL_TS_CSID,R3       ; Get CSID of system issuing timestamps
            29    13           BEQL    40$                     ; It's us
          049A    30           BSBW    LCK$ALLOC_LONGCDRP      ; Allocate a CDRP
        51 50     E9           BLBC    R0,70$                  ; Error
                                ;
                                ; Store necessary info. in CDRP to be able to start a deadlock
                                ; search later.
                                ;
    52   2C A5    9E           MOVAB   CDRP$L_VAL1(R5),R2      ; Point into CDRP data area
            54    10           BSBB    80$                     ; Store data in CDRP
 5C A5     50    D0           MOVL    R0,CDRP$L_VAL9(R5)      ; Store victim CSID
       30 A4     D0           MOVL    LKB$L_LKID(R4),-        ; Store next lockid
          60 A5                       CDRP$L_VAL10(R5)
      0270'CF    9E           MOVAB   W^BLD_TIMESTAMP_RQST,-  ; Store address of message build routine
          4C A5                       CDRP$L_MSGBLD(R5)
```

```
                                                H 13
                     002B   214
                     002B   215          ; Remove lock from timeout queue, reset the stack, and send the message.
                     002B   216
          50   64    0F   002B   217          REMQUE  LKB$L_ASTQFL(R4),R0     ; Remove LKB from timeout queue
        0040 8F    AA   002E   218          BICW    #LKB$M_TIMOUTQ,-        ; Clear corresponding status bit
           2A A4         0032   219                  LKB$W_STATUS(R4)
             0443    30   0034   220          BSBW    SEND_DLCK_MSG           ; Send the message
                2E   11   0037   221          BRB     70$
                     0039   222
                     0039   223 40$:       ; We are issuing timestamps
                     0039   224
          55   54    D0   0039   225          MOVL    R4,R5                   ; Save LKB address
        54   4E A4    9A   003C   226          MOVZBL  LKB$B_TSLT(R4),R4       ; Get timestamp lifetime
                6D    10   0040   227          BSBB    GET_TIMESTAMP
             22 54    E9   0042   228          BLBC    R4,70$                  ; Bitmap in use
          54   55    D0   0045   229          MOVL    R5,R4                   ; Restore LKB address
      59  0018'CF    9E   0048   230          MOVAB   W^LKMSG_BFR,R9          ; Point to internal message buffer
           E8 A9     D5   004D   231          TSTL    -FKB$K_LENGTH(R9)       ; Is it in use?
             15      12   0050   232          BNEQ    70$                     ; Yes
  08 A9    0A02 8F    B0   0052   233          MOVW    #LKMSG$K_SRCHDLCK@8-    ; Store facility and function codes
                          0058   234                  !CLSMSG$K_FAC_LCK,CLSMSG$B_FACILITY(R9)
        52   0C A9    9E   0058   235          MOVAB   LKMSG$W_MEMSEQ(R9),R2   ; Point to data area
                12    10   005C   236          BSBB    80$                     ; Fill in fields
        51   30 A4    D0   005E   237          MOVL    LKB$L_LKID(R4),R1       ; Get next lockid to search (this one)
        2C A9    50   7D   0062   238          MOVQ    R0,LKMSG$L_VCTMCSID(R9) ; Store victim CSID and next lockid
                05    05   0066   239          RSB                             ; Return to caller
                     0067   240
                     0067   241 70$:       ; This exit unwinds the stack and exits deadlock detection.
                     0067   242          ; If we sent a message then the original lock has been removed
                     0067   243          ; from the timeout queue.  We want to exit deadlock detection rather
                     0067   244          ; than trying to search for another deadlock because we will be
                     0067   245          ; unable to get another timestamp.  If we were unable to allocate a CDRP
                     0067   246          ; or the bitmap was in use then we leave the lock on the timeout queue
                     0067   247          ; so that we will retry this operation 1 second from now.
                     0067   248
             5E   5A    D0   0067   249          MOVL    R10,SP                  ; Reset stack
  00000000'GF     17   006A   250          JMP     G^LCK$DLCKEXIT          ; Return
                     0070   251
                     0070   252 ;
                     0070   253 ; Local subroutine to store message data in either CDRP or internal
                     0070   254 ; message template
                     0070   255 ;
                     0070   256 ;       Inputs: R0,R1   Timestamp expiration (if assigned locally)
                     0070   257 ;               R2      Address of data area in CDRP or internal message bfr
                     0070   258 ;               R4      Address of original LKB
                     0070   259 ;               R8      Original EPID
                     0070   260 ;
                     0070   261 ;       Outputs:
                     0070   262 ;               R0      Victim CSID (not stored in data area because
                     0070   263 ;                       CDRP$L_VAL9 is not contiguous with CDRP$L_VAL8.)
                     0070   264 ;
                     0070   265
             55   DD    0070   266 80$:       PUSHL   R5
  55   00000000'GF     0072   267          MOVL    G^CLU$GL_CLUB,R5        ; Get address of CLUB
    82   00AC C5       0079   268          MOVW    CLUB$W_MEMSEQ(R5),(R2)+ ; Store memseq
      82   4E A4       007E   269          MOVZBW  LKB$B_TSLT(R4),(R2)+    ; Store timestamp lifetime
          82   58       0082   270          MOVL    R8,(R2)+                ; Store original EPID
```

```
                82    30 A4    D0  0085   271          MOVL    LKB$L_LKID(R4),(R2)+    ; Store original lockid
                82    60 A5    D0  0089   272          MOVL    CLUB$L_LOCAL_CSID(R5),(R2)+ ; Store CSID of this system
                   82    50    7D  008D   273          MOVQ    R0,(R2)+                ; Store timestamp
                   82    01    CE  0090   274          MNEGL   #1,(R2)+                ; Initialize victim priority
                51    30 A4    D0  0093   275          MOVL    LKB$L_LKID(R4),R1       ; Get local lockid
                50    60 A5    D0  0097   276          MOVL    CLUB$L_LOCAL_CSID(R5),R0; and local CSID
                         04    E1  009B   277          BBC     #LKB$V_MSTCPY,-         ; Branch if not master copy
                      08 2A A4        009D   278                  LKB$W_STATUS(R4),85$
                51    54 A4    D0  00A0   279          MOVL    LKB$L_REMLKID(R4),R1    ; Get remote lockid instead
                50    58 A4    D0  00A4   280          MOVL    LKB$L_CSID(R4),R0       ; and remote CSID
                   82    51    D0  00A8   281 85$:     MOVL    R1,(R2)+                ; Store victim lockid; return victim
                         55 BED0  00AB   282          POPL    R5                      ; CSID in R0
                               05  00AE   283          RSB
                                   00AF   284
```

```
                                00AF   286          .SBTTL  GET_TIMESTAMP - Get a bitmap time stamp
                                00AF   287
                                00AF   288 ;++
                                00AF   289 ; FUNCTIONAL DESCRIPTION:
                                00AF   290 ;
                                00AF   291 ;       This routine returns a bitmap timestamp with a specified lifetime.
                                00AF   292 ;       The timestamp lifetime is encoded as a shift count that represents
                                00AF   293 ;       the number of bits the basic lifetime should be shifted.
                                00AF   294 ;       For example, if the basic lifetime unit is 50 ms. (TSLT_UNITS) then
                                00AF   295 ;       a specified lifetime (R4) of 2 would return a timestamp with
                                00AF   296 ;       an expiration time 200 ms. from now.
                                00AF   297 ;
                                00AF   298 ; CALLING SEQUENCE:
                                00AF   299 ;
                                00AF   300 ;       BSBW    GET_TIMESTAMP
                                00AF   301 ;
                                00AF   302 ; INPUT PARAMETERS:
                                00AF   303 ;
                                00AF   304 ;       R4      Timestamp lifetime (encoded as a shift count)
                                00AF   305 ;
                                00AF   306 ;
                                00AF   307 ; OUTPUT PARAMETERS:
                                00AF   308 ;
                                00AF   309 ;       R0,R1   Quadword expiration time (success only)
                                00AF   310 ;       R4      Completion code:  0 = failure
                                00AF   311 ;                                 1 = success
                                00AF   312 ; SIDE EFFECTS:
                                00AF   313 ;
                                00AF   314 ;       On success, bitmap is cleared, new expiration time is
                                00AF   315 ;       stored as both local and exact expiration time
                                00AF   316 ;--
                                00AF   317
                                00AF   318 GET_TIMESTAMP:
                                00AF   319           ; Determine if the previous timestamp has expired yet.
                                00AF   320           ; Note that normally, this test should be performed at the IPL
                                00AF   321           ; of the hardware clock interrupt (IPL$_HWCLK).  However, we can
                                00AF   322           ; tolerate the race condition here.  The result would be to think
                                00AF   323           ; that the bitmap is in use when it really wasn't.  If this occurs,
                                00AF   324           ; we will simply try again later.
                                00AF   325
                     2C   BB    00AF   326           PUSHR   #^M<R2,R3,R5>
52   00000000'GF      7E    00B1   327           MOVAQ   G^LCK$GQ_BITMAP_EXP,R2  ; Get address of expiration time
50   00000000'GF      7E    00B8   328           MOVAQ   G^EXE$GQ_SYSTIME,R0     ; Get address of system time
   04 A0    04 A2    D1    00BF   329           CMPL    4(R2),4(R0)             ; Compare low order time
                     0C   1F    00C4   330           BLSSU   20$                     ; Bitmap is available
                     05   1A    00C6   331           BGTRU   10$                     ; Bitmap is in use
            60       62   D1    00C8   332           CMPL    (R2),(R0)               ; Compare high order time
                     05   18    00CB   333           BLEQU   20$                     ; Bitmap is available
                                00CD   334
                                00CD   335 10$:      ; Bitmap is in use.  Return failure.
                                00CD   336
                     54   D4    00CD   337           CLRL    R4
                     2C   BA    00CF   338           POPR    #^M<R2,R3,R5>
                     05   00D1   339           RSB
                                00D2   340
                                00D2   341 20$:      ; Bitmap is available.  Compute new expiration times and clear bitmap.
                                00D2   342
```

```
                54    0007A120 8F    54    78  00D2    343          ASHL    R4,#TSLT_UNITS,R4       ; Compute time stamp lifetime
                               50    60    7D  00DA    344          MOVQ    (R0),R0                 ; Get current time
                               50    54    C0  00DD    345          ADDL    R4,R0                   ; Compute expiration time
                               51    00    D8  00E0    346          ADWC    #0,R1
                               62    50    7D  00E3    347          MOVQ    R0,(R2)                 ; Store expiration time
                           08  A2    50    7D  00E6    348          MOVQ    R0,8(R2)                ; Store local expiration time
                               7E    50    7D  00EA    349          MOVQ    R0,-(SP)                ; Save timestamp for return to caller
                     50    00000000'GF    D0  00ED    350          MOVL    G^LCK$GL_PRCMAP,R0      ; Get address of bitmap
           60    F8 A0    00    60    00    2C  00F4    351          MOVC5   #0,(R0),#0,-8(R0),(R0)  ; Clear it
                               54    01    D0  00FB    352          MOVL    #1,R4                   ; Return success
                                     2F    BA  00FE    353          POPR    #^M<R0,R1,R2,R3,R5>
                                           05  0100    354          RSB
```

```
                     0101   356          .SBTTL  LCK$RCV_TIMESTAMP_RQST
                     0101   357
                     0101   358 ;++
                     0101   359 ; FUNCTIONAL DESCRIPTION:
                     0101   360 ;
                     0101   361 ;       This routine is called by the received message dispatcher when
                     0101   362 ;       we receive a request for a timestamp.  If we can assign a timestamp
                     0101   363 ;       then we send a message that starts the deadlock search.  If we
                     0101   364 ;       cannot assign a timestamp (because the previous one has not
                     0101   365 ;       expired yet), then we send a message that will cause the original
                     0101   366 ;       lock to be requeued to the timeout queue.
                     0101   367 ;
                     0101   368 ; CALLING SEQUENCE:
                     0101   369 ;
                     0101   370 ;       JSB     LCK$RCV_TIMESTAMP_RQST  (called by received message dispatcher)
                     0101   371 ;
                     0101   372 ; INPUT PARAMETERS:
                     0101   373 ;
                     0101   374 ;       R2      Address of message buffer
                     0101   375 ;       R3      Address of CSB
                     0101   376 ;
                     0101   377 ; OUTPUT PARAMETERS:
                     0101   378 ;
                     0101   379 ;       None
                     0101   380 ;
                     0101   381 ; SIDE EFFECTS:
                     0101   382 ;
                     0101   383 ;       R0 - R5 not preserved
                     0101   384 ;
                     0101   385 ;       If a timestamp is assigned, the bitmap is cleared and the new
                     0101   386 ;       expiration time is stored as both the local and exact expiration time.
                     0101   387 ;--
                     0101   388
                     0101   389 LCK$RCV_TIMESTAMP_RQST::
          0388   30  0101   390          BSBW    RCV_DLCK_MSG
            0C   BB  0104   391          PUSHR   #^M<R2,R3>
     0000'CF   D5  0106   392          TSTL    W^LCK$GL_TS_CSID        ; Verify we are assigning timestamps
            46   12  010A   393          BNEQ    70$                     ; Error!
                     010C   394
                     010C   395          ; Get a timestamp
                     010C   396
     54 0E A2   9A  010C   397          MOVZBL  LKMSG$B_TSLT(R2),R4     ; Get timestamp lifetime
            9D   10  0110   398          BSBB    GET_TIMESTAMP
         05 54   E8  0112   399          BLBS    R4,20$                  ; Success
          02A3   30  0115   400          BSBW    LCK$SND_REDO_SRCH       ; Failure - redo deadlock search
            33   11  0118   401          BRB     50$
                     011A   402
                     011A   403 20$:     ; Have a timestamp.  Send a message that will initiate the deadlock
                     011A   404          ; search.  Store all necessary fields in the CDRP.
                     011A   405
     53   50   7D  011A   406          MOVQ    R0,R3                   ; Move timestamp
          038D   30  011D   407          BSBW    LCK$ALLOC_LONGCDRP      ; Allocate a CDRP
         05 50   E8  0120   408          BLBS    R0,30$                  ; Have one
          03A5   30  0123   409          BSBW    WAIT_FOR_POOL
            25   11  0126   410          BRB     50$
                     0128   411
     0278'CF   9E  0128   412 30$:     MOVAB   W^BLD_SRCHDLCK,-        ; Store address of message build routine
```

```
                4C A5          012C   413                  CDRP$L_MSGBLD(R5)
                0C A2    7D    012E   414          MOVQ    LKMSG$Q_MEMSEQ(R2),-    ; Store memseq, timestamp lifetime,
                2C A5          0131   415                  CDRP$L_VAL1(R5)         ; and original EPID
                14 A2    7D    0133   416          MOVQ    LKMSG$L_ORIGLKID(R2),-  ; Store original lockid and CSID
                34 A5          0136   417                  CDRP$L_VAL3(R5)
          3C A5    53    7D    0138   418          MOVQ    R3,CDRP$L_VAL5(R5)      ; Store timestamp
                24 A2    7D    013C   419          MOVQ    LKMSG$L_VCTMPRI(R2),-   ; Store deadlock victim priority
                44 A5          013F   420                  CDRP$L_VAL7(R5)         ; and lockid
                2C A2    7D    0141   421          MOVQ    LKMSG$L_VCTMCSID(R2),-  ; Store deadlock victim CSID and
                5C A5          0144   422                  CDRP$L_VAL9(R5)         ; next lockid
                               0146   423
                               0146   424          ; Send the message
                               0146   425
          53    18 A2    D0    0146   426          MOVL    LKMSG$L_ORIGCSID(R2),R3 ; Get CSID of original system
                032D     30    014A   427          BSBW    SEND_DLCK_MSG
                               014D   428
                               014D   429 50$:      ; Deallocate the message buffer and return
                               014D   430
                   0C    BA    014D   431          POPR    #^M<R2,R3>
                034D     31    014F   432          BRW     DEALL_DLCK_MSG
                               0152   433
                               0152   434
                               0152   435 70$:      BUG_CHECK       LOCKMGRERR,FATAL; This system is not issuing timestamps
```

```
                                0156   437          .SBTTL  CHECK_TIMESTAMP - Check bitmap timestamp
                                0156   438
                                0156   439 ;++
                                0156   440 ; FUNCTIONAL DESCRIPTION:
                                0156   441 ;
                                0156   442 ;       This routine is called when an incoming deadlock search message
                                0156   443 ;       arrives and needs to use this system's bitmap.  If the
                                0156   444 ;       expiration timestamp in the message is newer (greater than)
                                0156   445 ;       than the timestamp for this system's bitmap, then the bitmap
                                0156   446 ;       is cleared and the newer timestamp is stored.  If they are equal,
                                0156   447 ;       then the bitmap can be used immediately.  If the timestamp
                                0156   448 ;       in the message is older than the one for the bitmap, then this
                                0156   449 ;       indicates that the bitmap has been preempted by a newer request
                                0156   450 ;       and therefore this deadlock search is aborted for now, but
                                0156   451 ;       retried later, most likely with a timestamp with a longer lifetime.
                                0156   452 ;
                                0156   453 ; CALLING SEQUENCE:
                                0156   454 ;
                                0156   455 ;       BSBW    CHECK_TIMESTAMP
                                0156   456 ;
                                0156   457 ; INPUT PARAMETERS:
                                0156   458 ;
                                0156   459 ;       R2      Address of message buffer
                                0156   460 ;
                                0156   461 ; OUTPUT PARAMETERS:
                                0156   462 ;
                                0156   463 ;       R0      Completion code:  0 = abort deadlock search
                                0156   464 ;                                 1 = use bitmap
                                0156   465 ;       R7      Address of bitmap
                                0156   466 ;
                                0156   467 ; SIDE EFFECTS:
                                0156   468 ;
                                0156   469 ;       If the bitmap is cleared, then the local expiration time
                                0156   470 ;       is reset.
                                0156   471 ;       R1 is not preserved
                                0156   472 ;--
                                0156   473
                                0156   474 CHECK_TIMESTAMP:
57   00000000'GF    D0          0156   475         MOVL    G^LCK$GL_PRCMAP,R7      ; Get address of bitmap
51   00000000'GF    7E          015D   476         MOVAQ   G^LCK$GQ_BITMAP_EXP,R1  ; Get address of bitmap timestamp
   20 A2    04 A1    D1          0164   477         CMPL    4(R1),LKMSG$Q_BITMAP_EXP+4(R2) ; Compare high order times
                0A    1F          0169   478         BLSSU   10$                     ; Reuse bitmap
                30    1A          016B   479         BGTRU   40$                     ; Bitmap has been preempted
      1C A2     61    D1          016D   480         CMPL    (R1),LKMSG$Q_BITMAP_EXP(R2) ; Compare low order times
                2A    1A          0171   481         BGTRU   40$                     ; Bitmap has been preempted
                24    13          0173   482         BEQL    20$                     ; Continue using bitmap
                                0175   483
                                0175   484 10$:      ; Bitmap may be used after it is initialized.  Store new timestamps.
                                0175   485         ; The expiration timestamp is the one in the message.  The local
                                0175   486         ; timestamp is the current system time plus the maximum timestamp
                                0175   487         ; lifetime.
                                0175   488
      61    1C A2     7D          0175   489         MOVQ    LKMSG$Q_BITMAP_EXP(R2),(R1) ; Store new expiration timestamp
                3C    BB          0179   490         PUSHR   #^M<R2,R3,R4,R5>
52   00000000'GF    7D          017B   491         MOVQ    G^EXE$GQ_SYSTIME,R2     ; Get this system's time
52   00F42400 8F    C0          0182   492         ADDL    #<TSLT_UNITS*MAX_TSLT>,R2; Add maximum timestamp lifetime
        53    00    D8          0189   493         ADWC    #0,R3
```

```
                    08 A1   52   7D   018C   494          MOVQ    R2,8(R1)                        ; Store local expiration time
    67   F8 A7    00  67   00   2C   0190   495          MOVC5   #0,(R7),#0,-8(R7),(R7)          ; Clear it
                         3C   BA   0197   496          POPR    #^M<R2,R3,R4,R5>
                    50   01   D0   0199   497 20$:      MOVL    #1,R0
                              05   019C   498          RSB
                              019D   499
                    50   D4   019D   500 40$:      CLRL    R0                              ; Bitmap was preempted
                              05   019F   501          RSB
```

```
                     01A0   503          .SBTTL  LCK$SND_SRCHDLCK - Send deadlock search message
                     01A0   504 ;++
                     01A0   505 ; FUNCTIONAL DESCRIPTION:
                     01A0   506 ;
                     01A0   507 ;       This routine sends a search for deadlock message when either
                     01A0   508 ;       a master copy lock is blocking another lock or a waiting
                     01A0   509 ;       lock is found that is mastered on another system.  In effect,
                     01A0   510 ;       this message serves to "follow an edge" of the "wait-for" graph.
                     01A0   511 ;
                     01A0   512 ; CALLING SEQUENCE:
                     01A0   513 ;
                     01A0   514 ;       BSBW    LCK$SND_SRCHDLCK
                     01A0   515 ;       Note:   This routine may not return to its caller if called
                     01A0   516 ;               without a timestamp assigned (R9=0).  In this case,
                     01A0   517 ;               a message requesting a timestamp is sent and the stack
                     01A0   518 ;               is unwound and we exit the deadlock search.
                     01A0   519 ;               Also, if we fail to allocate a CDRP, we also unwind the
                     01A0   520 ;               stack and exit the deadlock search.
                     01A0   521 ;
                     01A0   522 ; INPUT PARAMETERS:
                     01A0   523 ;
                     01A0   524 ;       R6      Address of LKB
                     01A0   525 ;       R9      Address of Message buffer or 0 indicating none
                     01A0   526 ;       R10     Bottom of stack
                     01A0   527 ;
                     01A0   528 ; IMPLICIT INPUTS:
                     01A0   529 ;
                     01A0   530 ;       The region of stack bounded by R10 and SP contains a series
                     01A0   531 ;       of stack frames that describe that current "wait-for" cycle
                     01A0   532 ;       (see description below)
                     01A0   533 ;
                     01A0   534 ; OUTPUT PARAMETERS:
                     01A0   535 ;
                     01A0   536 ;       R0      Completion code:
                     01A0   537 ;               0 = exit normally
                     01A0   538 ;               -1 = exit due to failure to allocate a CDRP; stack
                     01A0   539 ;                    has been unwound back to original caller.
                     01A0   540 ;       R9      Address of message buffer if timestamp assigned
                     01A0   541 ;
                     01A0   542 ; SIDE EFFECTS:
                     01A0   543 ;
                     01A0   544 ;       R1 not preserved
                     01A0   545 ;--
                     01A0   546
                     01A0   547 LCK$SND_SRCHDLCK::
 007C 8F    BB       01A0   548          PUSHR   #^M<R2,R3,R4,R5,R6>     ; Can't change this without also
                     01A4   549                                          ; changing value of LOCKFRAME and
                     01A4   550                                          ; deadlock resolution code
                     01A4   551
                     01A4   552          ; Determine if a timestamp has been assigned.  R9 = 0 indicates
                     01A4   553          ; none was assigned.  R9 <> 0 indicates it points
                     01A4   554          ; to a message buffer and therefore, a timestamp has been assigned.
                     01A4   555
       59    D5      01A4   556          TSTL    R9                      ; Is there a timestamp assigned?
       03    12      01A6   557          BNEQ    10$                     ; Yes
     FE55    30      01A8   558          BSBW    LCK$SND_TIMESTAMP_RQST  ; No, get one (may not return here)
                     01AB   559
```

```
                     01AB   560 10$:    ; Allocate a CDRP
                     01AB   561
                     01AB   562          BSBW    LCK$ALLOC_LONGCDRP
                     01AE   563          BLBS    R0,15$                  ; Have one
                     01B1   564          MOVL    R9,R2                   ; Move address of message buffer
                     01B4   565          BSBW    WAIT_FOR_POOL
                     01B7   566          SUBL3   #4,RT0,SP               ; Unwind stack
                     01BB   567          MNEGL   #1,R0                   ; Set completion code
                     01BE   568          RSB                             ; Return to original caller
                     01BF   569
                     01BF   570 15$:
                     01BF   571 ;
                     01BF   572 ;        The stack consists of a series of stack frames, one for each lock
                     01BF   573 ;        involved in the current wait-for cycle.  Each stack frame consists
                     01BF   574 ;        of the 5 saved registers (R2 - R6) and a return address.  Note
                     01BF   575 ;        that in each stack frame the saved R6 points to the lock and the
                     01BF   576 ;        saved R4 points to the respective PCB lock queue (if the lock
                     01BF   577 ;        is not master copy).  Only the first and last frames should contain
                     01BF   578 ;        master copy locks.
                     01BF   579 ;        The stack frames are bounded by R10 and the current SP.
                     01BF   580 ;        The following diagram shows the stack with three frames.
                     01BF   581 ;
                     01BF   582 ;
                     01BF   583 ;          SP -> +---------------+           .     +-----------------------+
                     01BF   584 ;                |---------------|         .       |       Saved R2        |
                     01BF   585 ;                |---------------|       .         +-----------------------+
                     01BF   586 ;                |---------------|     .           |       Saved R3        |
                     01BF   587 ;                |---------------|   .             +-----------------------+
                     01BF   588 ;                |---------------| .               |       Saved R4        |
                     01BF   589 ;                +---------------+.                |  (PCB + PCB$L_LOCKQFL)|
                     01BF   590 ;                |---------------|                 +-----------------------+
                     01BF   591 ;                |---------------|                 |       Saved R5        |
                     01BF   592 ;                |---------------|                 +-----------------------+
                     01BF   593 ;                |---------------|                 |       Saved R6        |
                     01BF   594 ;                |---------------|.                |   (Address of LKB)    |
                     01BF   595 ;                +---------------+  .              +-----------------------+
                     01BF   596 ;                |---------------|    .            |    Return Address     |
                     01BF   597 ;                |---------------|      .          +-----------------------+
                     01BF   598 ;                |---------------|        .        |                       |
                     01BF   599 ;                |---------------|          .      +-----------------------+
                     01BF   600 ;                |---------------|            .
                     01BF   601 ;                +---------------+              .
                     01BF   602 ;          R10 ->
                     01BF   603 ;
                     01BF   604 ;        We will now search the frames looking for the process with
                     01BF   605 ;        the smallest deadlock priority.  When found, the respective
                     01BF   606 ;        deadlock priority will be compared with that in the input message.
                     01BF   607 ;        The objective is to find the best candidate for a deadlock victim
                     01BF   608 ;        if a deadlock is later found.  This candidate will be included in the
                     01BF   609 ;        message we send to the other system.  Note that a deadlock priority
                     01BF   610 ;        of zero causes an immediate exit from the loop.  Register usage will be:
                     01BF   611 ;
                     01BF   612 ;                R0      Current deadlock priority
                     01BF   613 ;                R1      Current lock frame pointer
                     01BF   614 ;                R2      Minimum deadlock priority, so far
                     01BF   615 ;                R3      Best victim frame, so far
                     01BF   616 ;                R4      Address of PCB lock queue (current frame)
```

```
                                                E 14
                     01BF   617 ;               R5      Address of CDRP
                     01BF   618 ;               R9      Address of message buffer
                     01BF   619 ;               R10     Bottom of stack (start search here)
                     01BF   620 ;               SP      Top of stack (end search here)
                     01BF   621 ;
                     01BF   622 ;       Note that the following code makes a number of assumptions
                     01BF   623 ;       regarding the order of registers saved on the stack and their
                     01BF   624 ;       contents.
                     01BF   625
        24 A9   D5   01BF   626         TSTL    LKMSG$L_VCTMPRI(R9)     ; Don't bother searching if the priority
           62   13   01C2   627         BEQL    50$                     ; in the message is zero
                     01C4   628
51 5A   18   C3   01C4   629         SUBL3   #LOCKFRAME,R10,R1       ; Initialize current frame pointer
   53   51   D0   01C8   630         MOVL    R1,R3                   ; Initialize "best" frame pointer
   52   01   CE   01CB   631         MNEGL   #1,R2                   ; Initialize "best" deadlock priority
50 10   A1   D0   01CE   632 20$:    MOVL    16(R1),R0               ; Get LKB address
        04   E1   01D2   633         BBC     #LKB$V_MSTCPY,-         ; Branch if not master copy
06 2A   A0        01D4   634                 LKB$W_STATUS(R0),25$
50 24   A0   D0   01D7   635         MOVL    LKB$L_DLCKPRI(R0),R0    ; Get deadlock priority from master copy
           08   11   01DB   636         BRB     28$
54 08   A1   D0   01DD   637 25$:    MOVL    8(R1),R4                ; Get pointer to PCB lock queue
50 08   A4   D0   01E1   638         MOVL    PCB$L_DLCKPRI-PCB$L_LOCKQFL(R4),R0 ; Get current deadlock pri.
           12   13   01E5   639 28$:    BEQL    35$                     ; Branch if zero - have best victim
   52   50   D1   01E7   640         CMPL    R0,R2                   ; Compare current priority with
           03   1E   01EA   641         BGEQU   30$                     ; previous minimum.
   52   50   7D   01EC   642         MOVQ    R0,R2                   ; This frame becomes "best so far"
   51   18   C2   01EF   643 30$:    SUBL    #LOCKFRAME,R1           ; Move to next frame
   5E   51   D1   01F2   644         CMPL    R1,SP                   ; Reached top of stack yet?
           D7   1E   01F5   645         BGEQU   20$                     ; No, repeat for next frame
           03   11   01F7   646         BRB     40$
   52   50   7D   01F9   647 35$:    MOVQ    R0,R2                   ; Move priority and frame pointer
                     01FC   648
                     01FC   649 40$:    ; Compare lowest deadlock priority so far (R2) with that in the
                     01FC   650         ; input message and select the lower.  R3 contains address of "best"
                     01FC   651         ; frame.
                     01FC   652
   24 A9   52   D1   01FC   653         CMPL    R2,LKMSG$L_VCTMPRI(R9)  ; Compare priorities
           24   1A   0200   654         BGTRU   50$                     ; The one in the message was lower
                     0202   655
                     0202   656         ; The one on the stack was lower.  R3 points to relevant frame.
                     0202   657
   51   10   A3   D0   0202   658         MOVL    16(R3),R1               ; Get address of LKB
        04   E1   0206   659         BBC     #LKB$V_MSTCPY,-         ; Branch if not master copy
     0A 2A   A1        0208   660                 LKB$W_STATUS(R1),45$
   53   54   A1   D0   020B   661         MOVL    LKB$L_REMLKID(R1),R3    ; Get remote lockid
   54   58   A1   D0   020F   662         MOVL    LKB$L_CSID(R1),R4       ; and CSID
           19   11   0213   663         BRB     60$
   53   30   A1   D0   0215   664 45$:    MOVL    LKB$L_LKID(R1),R3       ; Get lockid
50 00000000'GF   D0   0219   665         MOVL    G^CLU$GL_CLUB,R0        ; Get address of CLUB
   54   60   A0   D0   0220   666         MOVL    CLUB$L_LOCAL_CSID(R0),R4; Get local CSID
           08   11   0224   667         BRB     60$
                     0226   668
                     0226   669 50$:    ; The one in the message was lower.
                     0226   670
   52   24 A9   7D   0226   671         MOVQ    LKMSG$L_VCTMPRI(R9),R2  ; Get victim priority and lockid
   54   2C A9   D0   022A   672         MOVL    LKMSG$L_VCTMCSID(R9),R4 ; and CSID
                     022E   673
```

```
                                022E    674 60$:    ; Store info. in CDRP
                                022E    675
                                022E    676         MOVQ    R2,CDRP$L_VAL7(R5)      ; Store victim priority and lockid
                                0232    677         MOVL    R4,CDRP$L_VAL9(R5)      ; and CSID
                                0236    678         MOVL    LKB$L_REMLKID(R6),-     ; and next lockid to continue search
                                0239    679                 CDRP$L_VAL10(R5)
                                023B    680         PUSHL   R5
                                023D    681         MOVC3   #24,LKMSG$W_MEMSEQ(R9),-; and other fields
                                0241    682                 CDRP$L_VAL1(R5)
                                0243    683         POPL    R5
                                0246    684         MOVAB   B^BLD_SRCHDLCK,-        ; Store address of message build routine
                                0249    685                 CDRP$L_MSGBLD(R5)
                                024B    686
                                024B    687         ; Send the message
                                024B    688
                                024B    689         BBC     #LKB$V_MSTCPY,-         ; Branch if not master copy
                                024D    690                 LKB$W_STATUS(R6),70$
                                0250    691         MOVL    LKB$L_CSID(R6),R3       ; Get CSID
                                0254    692         BRB     75$
                                0256    693 70$:    MOVL    LKB$L_RSB(R6),R0        ; Get RSB address
                                025A    694         MOVL    RSB$L_CSID(R0),R3       ; Get CSID
                                025E    695 75$:    BSBW    SEND_DLCK_MSG
                                0261    696         POPR    #^M<R2,R3,R4,R5,R6>
                                0265    697         CLRL    R0                      ; Set completion code
                                0267    698         RSB
                                0268    699 ;
                                0268    700 ; Message build routine
                                0268    701 ;
                                0268    702 ;       Inputs: R2      Address of message buffer
                                0268    703 ;               R5      Address of CDRP
                                0268    704 ;
                                0268    705
                                0268    706 BLD_REDO_SRCH:
08 A2   0C02 8F     B0          0268    707         MOVW    #LKMSG$K_REDO_SRCH@8-   ; Store facility and function codes
                                026E    708                 !CLSMSG$R_FAC_LCK,CLSMSG$B_FACILITY(R2)
                    0E      11  026E    709         BRB     BLD_COMMON
                                0270    710
                                0270    711 BLD_TIMESTAMP_RQST:
08 A2   0902 8F     B0          0270    712         MOVW    #LKMSG$K_TSRQST@8-      ; Store facility and function codes
                                0276    713                 !CLSMSG$R_FAC_LCK,CLSMSG$B_FACILITY(R2)
                    06      11  0276    714         BRB     BLD_COMMON
                                0278    715
                                0278    716 BLD_SRCHDLCK:
08 A2   0A02 8F     B0          0278    717         MOVW    #LKMSG$K_SRCHDLCK@8-    ; Store facility and function codes
                                027E    718                 !CLSMSG$R_FAC_LCK,CLSMSG$B_FACILITY(R2)
                                027E    719 BLD_COMMON:
                    3C      BB  027E    720         PUSHR   #^M<R2,R3,R4,R5>
2C A5               20      28  0280    721         MOVC3   #32,CDRP$L_VAL1(R5),-   ; Move data from CDRP to message buffer
                    0C A2       0284    722                 LKMSG$W_MEMSEQ(R2)
                    3C      BA  0286    723         POPR    #^M<R2,R3,R4,R5>
                    5C A5   7D  0288    724         MOVQ    CDRP$L_VAL9(R5),-
                    2C A2       028B    725                 LKMSG$L_VCTMCSID(R2)
                                028D    726         RSB
```

```
                     028E   728          .SBTTL  LCK$RCV_SRCHDLCK - Receive search deadlock message
                     028E   729
                     028E   730 ;++
                     028E   731 ; FUNCTIONAL DESCRIPTION:
                     028E   732 ;
                     028E   733 ;       This routine is called when we receive a deadlock search message
                     028E   734 ;       for either a lock mastered on this system or a waiting lock owned
                     028E   735 ;       by this system.  We continue searching using this lock as our
                     028E   736 ;       starting point.
                     028E   737 ;
                     028E   738 ; CALLING SEQUENCE:
                     028E   739 ;
                     028E   740 ;       BSBW    LCK$RCV_SRCHDLCK (called from input message dispatcher)
                     028E   741 ;
                     028E   742 ; INPUT PARAMETERS:
                     028E   743 ;
                     028E   744 ;       R2      Address of message buffer
                     028E   745 ;       R3      Address of CSB
                     028E   746 ;
                     028E   747 ; OUTPUT PARAMETERS:
                     028E   748 ;
                     028E   749 ;       None
                     028E   750 ;
                     028E   751 ; SIDE EFFECTS:
                     028E   752 ;
                     028E   753 ;       Other deadlock search messages may be sent to other systems.
                     028E   754 ;       R0 and R1 are not preserved.
                     028E   755 ;--
                     028E   756
                     028E   757 LCK$RCV_SRCHDLCK::
        01FB   30    028E   758          BSBW    RCV_DLCK_MSG
    0FFC 8F    BB    0291   759          PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
    59   52    D0    0295   760          MOVL    R2,R9                           ; Move address of input message
                     029B   761
                     029B   762          ; Get lockid of lock to start search with.  Convert to LKB address.
                     029B   763
54  30 A2      D0    029B   764          MOVL    LKMSG$L_NEXTLKID(R2),R4 ; Get lockid
      01B5     30    029C   765          BSBW    LCK$CVT_ID_TO_LKB               ; Convert to LKB address
    0C 50      E9    029F   766          BLBC    R0,10$                          ; No LKB found; ignore message
                     02A2   767
                     02A2   768          ; Check bitmap expiration timestamp before we start using bitmap
                     02A2   769
      FEB1     30    02A2   770          BSBW    CHECK_TIMESTAMP                 ; Returns address of bitmap in R7
    09 50      E8    02A5   771          BLBS    R0,20$                          ; We can use bitmap
                     02A8   772
                     02A8   773          ; Bitmap has been preempted by a later deadlock search.
                     02A8   774          ; Double the bitmap lifetime requested and send back a message to
                     02A8   775          ; redo the original deadlock search.
                     02A8   776
    0E A2      96    02A8   777          INCB    LKMSG$B_TSLT(R2)                ; This will double the lifetime
      010D     30    02AB   778          BSBW    LCK$SND_REDO_SRCH
      008E     31    02AE   779 10$:     BRW     70$
                     02B1   780
                     02B1   781 20$:     ; Set up registers
                     02B1   782
58  10 A2      D0    02B1   783          MOVL    LKMSG$L_ORIGEPID(R2),R8 ; Original EPID
    5A   5E    D0    02B5   784          MOVL    SP,R10                          ; Current stack position
```

```
                00000000'GF   C1   02B8   785          ADDL3   G^LCK$GL_EXTRASTK,-      ; Compute stack limit
5B              00000000'GF        02BE   786                  G^EXE$GL_INTSTKLM,R11
                   5B    18   C0   02C4   787          ADDL    #LOCKFRAME,R11
                52    0C   A6   3C   02C7   788          MOVZWL  LKB$L_PID(R6),R2         ; Get process index
                           0B   13   02CB   789          BEQL    25$                      ; Branch if master copy or system owned
51              00000000'GF   D0   02CD   790          MOVL    G^SCH$GL_PCBVEC,R1       ; Convert to PCB address
                  54    6142   D0   02D4   791          MOVL    (R1)[R2],R4
                                   02D8   792
                                   02D8   793 25$:      ; The way in which we resume the deadlock search depends on
                                   02D8   794           ; whether this lock is a master (or local) or process copy.
                                   02D8   795
                50    50   A6   D0   02D8   796          MOVL    LKB$L_RSB(R6),R0         ; Get RSB address
                         38   A0   D5   02DC   797          TSTL    RSB$L_CSID(R0)           ; Is lock mastered here?
                              53   13   02DF   798          BEQL    60$                      ; Yes
                                   02E1   799
                                   02E1   800           ; This is a process copy lock.  For each lock this procress
                                   02E1   801           ; has in either CONVERT or WAITING state, see who is blocking
                                   02E1   802           ; those locks.
                                   02E1   803
                     55    56   D0   02E1   804          MOVL    R6,R5                    ; Move LKB address
                                52   D5   02E4   805          TSTL    R2                       ; If process index is 0 then lock is
                                57   13   02E6   806          BEQL    70$                      ; system owned
                53 67         52   E2   02E8   807          BBSS    R2,(R7),70$              ; Br. if we've already done this process
             54    0104   C4   DE   02EC   808          MOVAL   PCB$L_LOCKQFL(R4),R4     ; Point to lock queue header
                56    04   A4   D0   02F1   809          MOVL    4(R4),R6                 ; Get last lock in list
                     54    56   D1   02F5   810 30$:     CMPL    R6,R4                    ; Reached end of list?
                           45   13   02F8   811          BEQL    70$                      ; Yes
                56    C0   A6   DE   02FA   812          MOVAL   -LKB$L_OWNQFL(R6),R6     ; Point to start of LKB
                     56    55   D1   02FE   813          CMPL    R5,R6                    ; Is this the one we have in R5?
                           2B   13   0301   814          BEQL    35$                      ; Yes, move on to next one
                                   0303   815          DISPATCH        LKB$B_STATE(R6),TYPE=B,PREFIX=LKB$K_,-
                                   0303   816                  <-
                                   0303   817                  <CONVERT,32$>,-
                                   0303   818                  <WAITING,32$>-
                                   0303   819                  >
                        30   11   030D   820          BRB     70$                      ; Exit for all other states
                        09   E0   030F   821 32$:     BBS     #LCK$V_NODLCKWT,-        ; Branch if this lock should not be
                  1A 28   A6        0311   822                  LKB$W_FLAGS(R6),35$      ; considered as waiting for other locks
                50    50   A6   D0   0314   823          MOVL    LKB$L_RSB(R6),R0         ; Get RSB for this lock
                      38   A0   D5   0318   824          TSTL    RSB$L_CSID(R0)           ; Is it managed elsewhere?
                           08   13   031B   825          BEQL    33$                      ; No
000001A0'       GF         16   031D   826          JSB     G^LCK$SND_SRCHDLCK       ; Yes, send a message to keep looking
                           06   11   0323   827          BRB     34$                      ; Continue on this PCB
00000000'       GF         16   0325   828 33$:     JSB     G^LCK$SRCH_RESDLCK       ; No, recursively search
                     11    50   E8   032B   829 34$:     BLBS    R0,70$                   ; If LBS, exit search
                56   44    A6   D0   032E   830 35$:     MOVL    LKB$L_OWNQBL(R6),R6      ; Get previous lock
                           C1   11   0332   831          BRB     30$                      ; Repeat
                                   0334   832
                                   0334   833 60$:     ; This lock is a local or master copy.  Just determine who is blocking
                                   0334   834           ; this lock after verifying that the lock is not granted.
                                   0334   835
                                   0334   836          ASSUME  LKB$K_GRANTED  GT  0
                        36   A6    95   0334   837          TSTB    LKB$B_STATE(R6)          ; Ignore message if lock is granted
                           06   14   0337   838          BGTR    70$
00000000'       GF         16   0339   839          JSB     G^LCK$SRCH_RESDLCK       ; Search for deadlock
                                   033F   840
                                   033F   841 70$:     ; Deallocate the original message buffer and exit
```

```
                     033F   842
OFFC 8F   BA 033F   843          POPR    #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
     0159  31 0343   844          BRW     DEALL_DLCK_MSG          ; Deallocate message buffer and return
```

```
                     0346   846          .SBTTL  LCK$SND_DLCKFND - Send deadlock found message
                     0346   847
                     0346   848 ;++
                     0346   849 ; FUNCTIONAL DESCRIPTION:
                     0346   850 ;
                     0346   851 ;       This routine sends a message informing another system that a
                     0346   852 ;       specified lock has been chosen as a deadlock victim.  The lock
                     0346   853 ;       on the destination system is either a local copy or a process
                     0346   854 ;       because it is on that system the the dequeue/cancel function
                     0346   855 ;       must be issued.
                     0346   856 ;
                     0346   857 ; CALLING SEQUENCE:
                     0346   858 ;
                     0346   859 ;       JSB     LCK$SND_DLCKFND
                     0346   860 ;       IPL must be a IPL$_SCS
                     0346   861 ;       Note:   If we don't have an input message and we are unable
                     0346   862 ;               to allocate a CDRP, then we will unwind the stack
                     0346   863 ;               and exit from deadlock detection.
                     0346   864 ;
                     0346   865 ; INPUT PARAMETERS:
                     0346   866 ;
                     0346   867 ;       R2      Lock id. of victim lock
                     0346   868 ;       R3      CSID of destination system
                     0346   869 ;       R9      Address of input message or 0 indicating no message
                     0346   870 ;       R10     Bottom of stack
                     0346   871 ;
                     0346   872 ; OUTPUT PARAMETERS:
                     0346   873 ;
                     0346   874 ;       R0      Completion code of -1 if we unwind the stack and exit
                     0346   875 ;               from deadlock searching.
                     0346   876 ;
                     0346   877 ; SIDE EFFECTS:
                     0346   878 ;
                     0346   879 ;       R0 - R5 not preserved
                     0346   880 ;--
                     0346   881
                     0346   882 LCK$SND_DLCKFND::
                     0346   883
          0164   30  0346   884          BSBW    LCK$ALLOC_LONGCDRP      ; Alloc. CDRP
        2D 50    E9  0349   885          BLBC    R0,80$                  ; Unable to allocate one
                     034C   886
                     034C   887          ; Store necessary information to build message in CDRP.
                     034C   888          ; The original lockid and CSID are stored only if we have
                     034C   889          ; an input message.  This will be used by the destination system
                     034C   890          ; to redo a deadlock search.  If we don't have an input message
                     034C   891          ; then this system will automatically redo the search.
                     034C   892
    50   00000000'GF   D0  034C   893          MOVL    G^CLU$GL_CLUB,R0
          00AC C0   B0  0353   894          MOVW    CLUB$W_MEMSEQ(R0),-     ; Store memseq.
              2C A5         0357   895                  CDRP$L_VAL1(R5)
              59 D5         0359   896          TSTL    R9                      ; Is there an input message?
              0C 13         035B   897          BEQL    10$                     ; No
          14 A9  7D         035D   898          MOVQ    LKMSG$L_ORIGLKID(R9),-  ; Yes, store original lockid and
              34 A5         0360   899                  CDRP$L_VAL3(R5)         ; CSID in CDRP
          0E A9  90         0362   900          MOVB    LKMSG$B_TSLT(R9),-      ; and timestamp lifetime
              2E A5         0365   901                  CDRP$L_VAL1+2(R5)
              03 11         0367   902          BRB     20$
```

```
                                ;
                                ; Action routine to build deadlock found message.  Inputs are:
                                ;
                                ;       R2      Address of message buffer
                                ;       R5      Address of CDRP
                                ;       CDRP$L_VAL1     MEMSEQ and timestamp lifetime
                                ;       CDRP$L_VAL3     Original lockid (or 0)
                                ;       CDRP$L_VAL4     Original CSID (or 0)
                                ;       CDRP$L_VAL5     Lockid of victim lock
                                ;       CDRP$L_VAL6     CSID of victim lock
                                ;
                                BLD_DLCKFND:
                                        ASSUME  CLSMSG$B_FUNC   EQ      1+CLSMSG$B_FACILITY
08 A2   0802 8F   B0            MOVW    #LKMSG$K_DLCKFND@8-             ; Store function and facility codes
                                        !CLSMSG$K_FAC_LCK,CLSMSG$B_FACILITY(R2)
2C A5   DO                      MOVL    CDRP$L_VAL1(R5),-               ; Store MEMSEQ and timestamp lifetime
0C A2                           LKMSG$Q_MEMSEQ(R2)
34 A5   7D                      MOVQ    CDRP$L_VAL3(R5),-               ; Store original lockid and CSID
14 A2                           LKMSG$L_ORIGLKID(R2)
3C A5   7D                      MOVQ    CDRP$L_VAL5(R5),-               ; Store victim lockid and CSID
28 A2                           LKMSG$L_VCTMLKID(R2)
        05              RSB
```

```
                        03A0   943          .SBTTL  LCK$RCV_DLCKFND - Receive deadlock found message
                        03A0   944
                        03A0   945 ;++
                        03A0   946 ; FUNCTIONAL DESCRIPTION:
                        03A0   947 ;
                        03A0   948 ;       This routine is called when we receive a deadlock found message.
                        03A0   949 ;       The message specifies a particular lock chosen to be a deadlock
                        03A0   950 ;       The lock must be either a local or process copy on this system.
                        03A0   951 ;
                        03A0   952 ; CALLING SEQUENCE:
                        03A0   953 ;
                        03A0   954 ;       JSB     LCK$RCV_DLCKFND  (called from received message dispatcher)
                        03A0   955 ;       IPL must be at IPL$_SCS
                        03A0   956 ;
                        03A0   957 ; INPUT PARAMETERS:
                        03A0   958 ;
                        03A0   959 ;       R2      Address of message buffer
                        03A0   960 ;       R3      Address of CSB
                        03A0   961 ;
                        03A0   962 ; OUTPUT PARAMETERS:
                        03A0   963 ;
                        03A0   964 ;       None
                        03A0   965 ;
                        03A0   966 ; SIDE EFFECTS:
                        03A0   967 ;
                        03A0   968 ;       R0 - R5 not preserved
                        03A0   969 ;--
                        03A0   970
                        03A0   971 LCK$RCV_DLCKFND::
            00E9    30  03A0   972          BSBW    RCV_DLCK_MSG
          03CC 8F   BB  03A3   973          PUSHR   #^M<R2,R3,R6,R7,R8,R9>
            59    52    D0  03A7   974          MOVL    R2,R9                   ; Move address of message
        52    28 A9   7D  03AA   975          MOVQ    LKMSG$L_VCTMLKID(R9),R2 ; Get victim lockid and CSID
00000000'GF     16  03AE   976          JSB     G^LCK$BREAK_DEADLOCK    ; Cancel the lock request
                        03B4   977
          03CC 8F   BA  03B4   978          POPR    #^M<R2,R3,R6,R7,R8,R9>
            00E4    31  03B8   979          BRW     DEALL_DLCK_MSG          ; Deallocate message buffer and return
```

```
                     03BB    981          .SBTTL  LCK$SND_REDO_SRCH - Send a redo deadlock search message
                     03BB    982
                     03BB    983 ;++
                     03BB    984 ; FUNCTIONAL DESCRIPTION:
                     03BB    985 ;
                     03BB    986 ;       This routine is called when it is necessary to redo a deadlock
                     03BB    987 ;       search.  It sends a message to the system mastering the lock
                     03BB    988 ;       unless that system is this system.
                     03BB    989 ;
                     03BB    990 ; CALLING SEQUENCE:
                     03BB    991 ;
                     03BB    992 ;       BSBW    LCK$SND_REDO_SRCH
                     03BB    993 ;
                     03BB    994 ; INPUT PARAMETERS:
                     03BB    995 ;
                     03BB    996 ;       R2      Address of message buffer
                     03BB    997 ;
                     03BB    998 ; IMPLICIT INPUTS:
                     03BB    999 ;
                     03BB   1000 ;       LKMSG$B_TSLT(R2)        Timestamp lifetime
                     03BB   1001 ;       LKMSG$L_ORIGLKID(R2)    Lockid of lock to repeat deadlock search
                     03BB   1002 ;       LKMSG$L_ORIGCSID(R2)    CSID of system mastering above lock
                     03BB   1003 ;                               (0 indicates original CSID and lockid
                     03BB   1004 ;                               are unknown, but it is not necessary
                     03BB   1005 ;                               to send the redo message - see LCK$SND_DLCKFND)
                     03BB   1006 ;
                     03BB   1007 ; OUTPUT PARAMETERS:
                     03BB   1008 ;
                     03BB   1009 ;       None
                     03BB   1010 ;
                     03BB   1011 ; SIDE EFFECTS:
                     03BB   1012 ;
                     03BB   1013 ;       R0 - R5 are not preserved
                     03BB   1014 ;--
                     03BB   1015
                     03BB   1016 LCK$SND_REDO_SRCH::
                     03BB   1017         ; Determine if the lock is mastered on this system
                     03BB   1018
      53    18 A2    D0   03BB   1019         MOVL    LKMSG$L_ORIGCSID(R2),R3 ; Get original CSID
            10       13   03BF   1020         BEQL    10$                     ; Not present
50  00000000'GF      D0   03C1   1021         MOVL    G^CLU$GL_CLUB,R0        ; Get address of CLUB
     60 A0    53     D1   03C8   1022         CMPL    R3,CLUB$L_LOCAL_CSID(R0); Is it the CSID of this system?
              04     12   03CC   1023         BNEQ    20$                     ; No
                          03CE   1024
                          03CE   1025         ; Lock is mastered on this system
                          03CE   1026
              0027   30   03CE   1027         BSBW    REDO_SRCH               ; Requeue it to the timeout queue
                     05   03D1   1028 10$:     RSB
                          03D2   1029
                          03D2   1030 20$:     ; Lock is mastered elsewhere
                          03D2   1031
              00D8   30   03D2   1032         BSBW    LCK$/_LOC_LONGCDRP      ; Allocate CDRP
           14 50     E9   03D5   1033         BLBC    R0,80$                  ; Unable to allocate one
           0C A2     D0   03D8   1034         MOVL    LKMSG$W_MEMSEQ(R2),-    ; Store MEMSEQ and timestamp lifetime
           2C A5                  03DB   1035                 CDRP$L_VAL1(R5)
           14 A2     D0   03DD   1036         MOVL    LKMSG$L_ORIGLKID(R2),-  ; Store lockid
           34 A5                  03E0   1037                 CDRP$L_VAL3(R5)
```

```
                                                     N 14
FE82 CF   9E  03E2  1038          MOVAB   W^BLD_REDO_SRCH,-       ; Store address of message build routine
  4C A5       03E6  1039                  CDRP$L_MSGBLD(R5)
   008F   30  03E8  1040          BSBW    SEND_DLCK_MSG           ; Send the message
          05  03EB  1041          RSB
              03EC  1042
   00DC   30  03EC  1043 80$:     BSBW    WAIT_FOR_POOL
          05  03EF  1044          RSB
```

```
                     03F0  1046          .SBTTL  LCK$RCV_REDO_SRCH
                     03F0  1047
                     03F0  1048 ;++
                     03F0  1049 ; FUNCTIONAL DESCRIPTION:
                     03F0  1050 ;
                     03F0  1051 ;       This routine is called when we receive a message to redo a deadlock
                     03F0  1052 ;       search for a lock mastered on this system.
                     03F0  1053 ;
                     03F0  1054 ; CALLING SEQUENCE:
                     03F0  1055 ;
                     03F0  1056 ;       BSBW    LCK$RCV_REDO_SRCH  (called from input message dispatcher)
                     03F0  1057 ;
                     03F0  1058 ; INPUT PARAMETERS:
                     03F0  1059 ;
                     03F0  1060 ;       R2      Address of message buffer
                     03F0  1061 ;       R3      Address of CSB
                     03F0  1062 ;
                     03F0  1063 ; OUTPUT PARAMETERS:
                     03F0  1064 ;
                     03F0  1065 ;       None
                     03F0  1066 ;
                     03F0  1067 ; SIDE EFFECTS:
                     03F0  1068 ;
                     03F0  1069 ;       R0, R1 and R4 not preserved
                     03F0  1070 ;--
                     03F0  1071
                     03F0  1072 LCK$RCV_REDO_SRCH::
 0099   30           03F0  1073          BSBW    RCV_DLCK_MSG
   03   10           03F3  1074          BSBB    REDO_SRCH                   ; Do the work
 00A7   31           03F5  1075          BRW     DEALC_DLCK_MSG              ; Deallocate message buffer and return
```

```
                                03F8  1077          .SBTTL  REDO_SRCH - Redo deadlock search
                                03F8  1078
                                03F8  1079 ;++
                                03F8  1080 ; FUNCTIONAL DESCRIPTION:
                                03F8  1081 ;
                                03F8  1082 ;       This routine is called to requeue a lock back on the timeout
                                03F8  1083 ;       queue when a deadlock search must be repeated.  Deadlock searches
                                03F8  1084 ;       are repeated for reasons such as:
                                03F8  1085 ;
                                03F8  1086 ;               o  One deadlock has already been found for this lock
                                03F8  1087 ;               o  A timestamp could not be issued
                                03F8  1088 ;               o  The deadlock search was incomplete for some reason
                                03F8  1089 ;                  (e.g. unable to allocate pool or our timestamp was
                                03F8  1090 ;                   superseded)
                                03F8  1091 ;
                                03F8  1092 ; CALLING SEQUENCE:
                                03F8  1093 ;
                                03F8  1094 ;       BSBW    REDO_SRCH
                                03F8  1095 ;
                                03F8  1096 ; INPUT PARAMETERS:
                                03F8  1097 ;
                                03F8  1098 ;       R2      Address of message buffer
                                03F8  1099 ;
                                03F8  1100 ; IMPLICIT INPUTS:
                                03F8  1101 ;
                                03F8  1102 ;       LKMSG$B_TSLT(R2)        Timestamp lifetime
                                03F8  1103 ;       LKMSG$L_ORIGLKID(R2)    Lockid of lock to repeat deadlock search
                                03F8  1104 ;
                                03F8  1105 ; OUTPUT PARAMETERS:
                                03F8  1106 ;
                                03F8  1107 ;       None
                                03F8  1108 ;
                                03F8  1109 ; SIDE EFFECTS:
                                03F8  1110 ;
                                03F8  1111 ;       R0, R1 and R4 not preserved
                                03F8  1112 ;--
                                03F8  1113
                                03F8  1114 REDO_SRCH:
                          56    DD  03F8  1115          PUSHL   R6                              ; Save R6
                    54    14 A2   DO  03FA  1116          MOVL    LKMSG$L_ORIGLKID(R2),R4 ; Get lockid
                        0053    30  03FE  1117          BSBW    LCK$CVT_ID_TO_LKB       ; Convert to LKB address
                      48    50    E9  0401  1118          BLBC    R0,60$                  ; No LKB
                    50    50 A6   DO  0404  1119          MOVL    LKB$L_RSB(R6),R0        ; Get RSB address
                      38    A0    D5  0408  1120          TSTL    RSB$L_CSID(R0)          ; Verify it's mastered here
                            43    12  040B  1121          BNEQ    90$                     ; Error!
                                      040D  1122
                                      040D  1123          ; Only requeue lock if it's in either CONVERT or WAITING state
                                      040D  1124          ; and it's not already queued.
                                      040D  1125
                                      040D  1126          DISPATCH        LKB$B_STATE(R6),TYPE=B,PREFIX=LKB$K_,-
                                      040D  1127          <-
                                      040D  1128          <CONVERT,30$>,-
                                      040D  1129          <WAITING,30$>,-
                                      040D  1130          >
                                33    11  0417  1131          BRB     60$                     ; Ignore for other states
                                      0419  1132
                                      0419  1133 30$:     ; If we haven't used up all the retries (MAX_TSLT) then the lock
```

```
                                0419  1134          ; is requeued at the front of the timeout queue.  If we have
                                0419  1135          ; used up all retries, then the retry count is cleared, and the
                                0419  1136          ; lock is queued at the back of the timeout queue with another
                                0419  1137          ; wait time applied.  The result is to retry a deadlock search for
                                0419  1138          ; this lock later.
                                0419  1139
                          06    0419  1140          BBSS    #LKB$V_TIMOUTQ,-        ; Branch if already on the queue;
                    2E 2A A6    041B  1141                  LKB$W_STATUS(R6),60$    ; set bit otherwise
          00000000'GF   D0      041E  1142          MOVL    G^EXE$GL_ABSTIM,-       ; Store immediate timeout time
                    18 A6       0424  1143                  LKB$L_DUETIME(R6)
               50   00000000'GF  DE  0426  1144          MOVAL   G^LCK$GL_TIMOUTQ,R0     ; Get address of timeout queue
                       0E A2    90   042D  1145          MOVB    LKMSG$B_TSLT(R2),-      ; Store timestamp lifetime
                       4E A6        0430  1146                  LKB$B_TSLT(R6)
                       4E A6    91   0432  1147          CMPB    LKB$B_TSLT(R6),-        ; Have we exceeded the maximum
                          05        0435  1148                  #MAX_TSLT               ; number of retries?
                          05    14   0436  1149          BGTR    40$                     ; Yes
                    60    66    0E   0438  1150          INSQUE  LKB$L_ASTQFL(R6),(R0)   ; No, insert at the head of the queue
                          0F    11   043B  1151          BRB     60$
                                043D  1152
                       4E A6    94   043D  1153  40$:    CLRB    LKB$B_TSLT(R6)          ; Reset retry count
                00000000'GF     C0   0440  1154          ADDL    G^LCK$GL_WAITTIME,-     ; Add another wait time to due time
                          18 A6       0446  1155                  LKB$L_DUETIME(R6)
                 04 B0    66    0E   0448  1156          INSQUE  LKB$L_ASTQFL(R6),@4(R0) ; Insert at the tail of the queue
                                044C  1157
                          56 8ED0   044C  1158  60$:    POPL    R6
                             05     044F  1159          RSB
                                0450  1160
                                0450  1161  90$:    BUG_CHECK       LOCKMGRERR,FATAL; Not mastered here
```

```
                                0454  1163          .SBTTL  LCK$CVT_ID_TO_LKB - Convert a lockid to LKB address
                                0454  1164
                                0454  1165 ;++
                                0454  1166 ; FUNCTIONAL DESCRIPTION:
                                0454  1167 ;
                                0454  1168 ;       This routine converts a lockid to a LKB address, if possible.
                                0454  1169 ;       The verification check of comparing remote lockids is not
                                0454  1170 ;       performed as both lockids are not available.
                                0454  1171 ;
                                0454  1172 ; CALLING SEQUENCE:
                                0454  1173 ;
                                0454  1174 ;       BSBW    LCK$CVT_ID_TO_LKB
                                0454  1175 ;
                                0454  1176 ; INPUT PARAMETERS:
                                0454  1177 ;
                                0454  1178 ;       R4      Lockid
                                0454  1179 ;
                                0454  1180 ; OUTPUT PARAMETERS:
                                0454  1181 ;
                                0454  1182 ;       R0      Completion code (0 = failure; 1 = success)
                                0454  1183 ;       R6      Address of LKB (success only)
                                0454  1184 ;
                                0454  1185 ; SIDE EFFECTS:
                                0454  1186 ;
                                0454  1187 ;       None
                                0454  1188 ;--
                                0454  1189
                                0454  1190
                                0454  1191 LCK$CVT_ID_TO_LKB::
          56    54    3C      0454  1192          MOVZWL  R4,R6                   ; Put lockid index in R6
00000000'GF     56    D1      0457  1193          CMPL    R6,G^LCK$GL_MAXID       ; Is the lock id too big?
                17    1A      045E  1194          BGTRU   60$                     ; Yes
50  00000000'GF     D0      0460  1195          MOVL    G^LCK$GL_IDTBL,R0       ; Get address of lockid table
          56  6046    D0      0467  1196          MOVL    (R0)[R6],R6             ; Get LKB address
                08    18      046B  1197          BGEQ    60$                     ; Unallocated id
        30 A6   54    D1      046D  1198          CMPL    R4,LKB$L_LKID(R6)       ; Check sequence number
                04    12      0471  1199          BNEQ    60$                     ; Not valid
          50    01    D0      0473  1200          MOVL    #1,R0
                05    05      0476  1201          RSB
                              0477  1202
                50    D4      0477  1203 60$:     CLRL    R0
                05    05      0479  1204          RSB
```

```
                                        .SBTTL  SEND_DLCK_MSG - Send any deadlock detection message

                                ;++
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ;       This routine is called to send any message when the caller
                                ;       wants control returned to it as opposed to it's caller.
                                ;       After the message has been acknowledged, the CDRP is deallocated.
                                ;       Note that all errors are ignored.
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       BSBW    SEND_DLCK_MSG
                                ;
                                ; INPUT PARAMETERS:
                                ;
                                ;       R3      CSID of destination system
                                ;       R5      Address of CDRP
                                ;
                                ; OUTPUT PARAMETERS:
                                ;
                                ;       None
                                ;
                                ; SIDE EFFECTS:
                                ;
                                ;       R0 - R2 and R4 are destroyed.
                                ;--

                                SEND_DLCK_MSG:
                                        .IF NE  CA$_MEASURE
                                        INCL    G^PMS$GL_DLCKMSGS_OUT
                                        .ENDC

                                        BSBW    CNX$SEND_MSG
                                        MOVL    R5,R0                   ; Address of CDRP
                                        JMP     G^EXE$DEANONPAGED       ; Deallocate it and return
```

```
                        048C  1243          .SBTTL  RCV_DLCK_MSG - Receive a deadlock message
                        048C  1244
                        048C  1245 ;++
                        048C  1246 ; FUNCTIONAL DESCRIPTION
                        048C  1247 ;
                        048C  1248 ;       This routine is called whenever we receive a deadlock message.
                        048C  1249 ;       Its purpose is to verify that the internal message buffer
                        048C  1250 ;       is available (i.e. not in use waiting for pool).  If it is
                        048C  1251 ;       in use, then we reject this message and break the connection.
                        048C  1252 ;
                        048C  1253 ; CALLING SEQUENCE:
                        048C  1254 ;
                        048C  1255 ;       BSBW    RCV_DLCK_MSG
                        048C  1256 ;       NOTE:   If we break the connection, then we return to our caller's
                        048C  1257 ;               caller, usually, the input message dispatcher.
                        048C  1258 ;
                        048C  1259 ; INPUT PARAMETERS:
                        048C  1260 ;
                        048C  1261 ;       R2      Address of message buffer
                        048C  1262 ;       R3      Address of CSB
                        048C  1263 ;
                        048C  1264 ; OUTPUT PARAMETERS:
                        048C  1265 ;
                        048C  1266 ;       None
                        048C  1267 ;
                        048C  1268 ; SIDE EFFECTS:
                        048C  1269 ;
                        048C  1270 ;       None if we return to our caller.
                        048C  1271 ;       The message buffer is deallocated if we break the connection.
                        048C  1272 ;--
                        048C  1273
                        048C  1274 RCV_DLCK_MSG:
0000'CF    D5           048C  1275          TSTL    W^LKMSG_FKB             ; Is fork block in use?
      07   12           0490  1276          BNEQ    10$                     ; Yes
                        0492  1277
        00000002        0492  1278          .IF NE  CA$_MEASURE
00000000'GF    D6       0492  1279          INCL    G^PMS$GL_DLCKMSGS_IN
                        0498  1280          .ENDC
                        0498  1281
                  05    0498  1282          RSB
                        0499  1283
      5E    04    C0    0499  1284 10$:     ADDL    #4,SP                   ; Pop caller's return address off stack
         FB61'    31    049C  1285          BRW     CNX$RCV_REJECT          ; Reject message
```

```
                        049F  1287          .SBTTL  DEALL_DLCK_MSG - Deallocate deadlock message buffer
                        049F  1288
                        049F  1289 ;++
                        049F  1290 ; FUNCTIONAL DESCRIPTION:
                        049F  1291 ;
                        049F  1292 ;       This routine is called to deallocate received deadlock message
                        049F  1293 ;       buffers.  However, it distinguishes between real message buffers
                        049F  1294 ;       and our internal buffer which is not deallocated.
                        049F  1295 ;
                        049F  1296 ; CALLING SEQUENCE:
                        049F  1297 ;
                        049F  1298 ;       BSBW    DEALL_DLCK_MSG
                        049F  1299 ;
                        049F  1300 ; INPUT PARAMETERS:
                        049F  1301 ;
                        049F  1302 ;       R2      Address of message buffer
                        049F  1303 ;       R5      Address of CSB
                        049F  1304 ;
                        049F  1305 ; OUTPUT PARAMETERS:
                        049F  1306 ;
                        049F  1307 ;       None
                        049F  1308 ;
                        049F  1309 ; SIDE EFFECTS:
                        049F  1310 ;
                        049F  1311 ;       R0 - R2 destroyed
                        049F  1312 ;--
                        049F  1313
                        049F  1314 DEALL_DLCK_MSG:
50   0018'CF    9E      049F  1315          MOVAB   W^LKMSG_BFR,R0          ; Get address of internal buffer
     52    50   D1      04A4  1316          CMPL    R0,R2                   ; Is it our internal message buffer?
           03   13      04A7  1317          BEQL    10$                     ; Yes
        FB54'   30      04A9  1318          BSBW    CNX$DEALL_MSG_BUF_CSB   ; No, deallocate real message buffer
                05      04AC  1319 10$:     RSB
```

```
                                        .SBTTL  LCK$ALLOC_LONGCDRP - Allocate a long CDRP

                                ;++
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ;       This routine is used to allocate a longer CDRP than is normally
                                ;       used for connection purposes.  The reason is because deadlock
                                ;       messages have more context than can fit into a regular sized
                                ;       CDRP.
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       BSBW    LCK$ALLOC_LONGCDRP
                                ;
                                ; INPUT PARAMETERS:
                                ;
                                ;       None
                                ;
                                ; OUTPUT PARAMETERS:
                                ;
                                ;       R0      Completion code
                                ;       R5      Address of CDRP
                                ;
                                ; COMPLETION CODES:
                                ;
                                ;       SS$_NORMAL      CDRP allocated
                                ;       SS$_INSFEM      Insufficient memory
                                ;
                                ; SIDE EFFECTS:
                                ;
                                ;       R0 and R1 not preserved.
                                ;--
                                ;
                                LCK$ALLOC_LONGCDRP:
          52    DD              POPSHL  R2
51    0064 8F    3C              MOVZWL  #CDRP$K_CM_LONG_LENGTH,R1; Size of CDRP
 00000000'GF     16              JSB     G^EXE$ALONONPAGED       ; Allocate pool
       0A 50     E9              BLBC    R0,80$                  ; Insufficient memory
    55    52     D0              MOVL    R2,R5                   ; Move address of CDRP
  08 A5    51     B0              MOVW    R1,CDRP$W_CDRPSIZE(R5)  ; Store size
        FB39'     30              BSBW    CNX$INIT_CDRP           ; Initialize CDRP
          52 8ED0   04C7  1362 80$:       POPL    R2
                05  04CA  1363           RSB
```

```
                                                J 15
04CB  1365          .SBTTL  WAIT_FOR_POOL - Wait for pool
04CB  1366
04CB  1367 ;++
04CB  1368 ; FUNCTIONAL DESCRIPTION:
04CB  1369 ;
04CB  1370 ;       This routine copies an input message into the internal
04CB  1371 ;       message buffer and does a FORK_WAIT.  Upon resumption,
04CB  1372 ;       we simulate receiving the same message again.  If another
04CB  1373 ;       input message is received while the fork block is queued,
04CB  1374 ;       then we reject the message and break the connection (see
04CB  1375 ;       RCV_DLCK_MSG).
04CB  1376 ;
04CB  1377 ; CALLING SEQUENCE:
04CB  1378 ;
04CB  1379 ;       BSBW    WAIT_FOR_POOL
04CB  1380 ;       NOTE:   If no input message is specified, then we unwind the
04CB  1381 ;               stack and return to LCK$DLCKEXIT.
04CB  1382 ;
04CB  1383 ; INPUT PARAMETERS:
04CB  1384 ;
04CB  1385 ;       R2      Address of input message (or 0 indicating no input message)
04CB  1386 ;       R10     Address of stack position to unwind to if R2=0
04CB  1387 ;
04CB  1388 ; OUTPUT PARAMETERS:
04CB  1389 ;
04CB  1390 ;       None
04CB  1391 ;
04CB  1392 ; SIDE EFFECTS:
04CB  1393 ;
04CB  1394 ;       R0 and R1 not preserved.
04CB  1395 ;       A fork block is queued that when resumed will call the input
04CB  1396 ;       message dispatcher.
04CB  1397 ;--
04CB  1398
04CB  1399 WAIT_FOR_POOL:
            52   D5  04CB  1400          TSTL    R2                      ; Do we have an input message?
            09   12  04CD  1401          BNEQ    10$                     ; Yes
         5E   5A   D0  04CF  1402          MOVL    R10,SP                  ; No, unwind stack
  00000000'GF   17  04D2  1403          JMP     G^LCK$DLCKEXIT          ; Exit deadlock detection
                    04D8  1404
      007C 8F   BB  04D8  1405 10$:     PUSHR   #^M<R2,R3,R4,R5,R6>     ; Save registers
56    0000'CF   9E  04DC  1406          MOVAB   W^LKMSG_FKB,R6          ; Get address of fork block
            66   D5  04E1  1407          TSTL    (R6)                    ; Verify it's not in use
            33   12  04E3  1408          BNEQ    90$                     ; Error
   55    18 A6   9E  04E5  1409          MOVAB   FKB$K_LENGTH(R6),R5     ; Get address of message buffer
65  62       34   28  04E9  1410          MOVC3   #LKMSG$K_DLM_LENGTH,(R2),(R5)   ; Copy message
      55   56   D0  04ED  1411          MOVL    R6,R5                   ; Move fork block address
            05   10  04F0  1412          BSBB    50$                     ; Queue fork block
      007C 8F   BA  04F2  1413          POPR    #^M<R2,R3,R4,R5,R6>
            05   04F6  1414          RSB
                    04F7  1415
                    04F7  1416 50$:     FORK_WAIT                       ; Fork and wait
            65   7C  04FD  1417          CLRQ    (R5)                    ; Indicate fork block is not is use
   52    18 A5   9E  04FF  1418          MOVAB   FKB$K_LENGTH(R5),R2     ; Get address of message buffer
            53   D4  0503  1419          CLRL    R3                      ; Indicate no CSB address
50  00000000'GF   D0  0505  1420          MOVL    G^CLU$GL_CLUB,R0        ; Get address of CLUB
      00AC C0   B1  050C  1421          CMPW    CLUB$W_MEMSEQ(R0),-     ; Has memseq changed?
```

```
                                                K 15
OC A2          0510  1422                     LKMSG$W_MEMSEQ(R2)
   03    12    0512  1423             BNEQ    60$                     ; Yes, ignore this message
 FAE9'   30    0514  1424             BSBW    LCK$DISPATCH            ; No, dispatch on this message
         05    0517  1425 60$:        RSB
               0518  1426
               0518  1427 90$:        BUG_CHECK       LOCKMGRERR,FATAL
               051C  1428
               051C  1429
               051C  1430
               051C  1431
               051C  1432             .END
```

```
$$BASE                          = FFFFFFFF
$$DISPL                         = 00000001
$$GENSW                         = 00000001
$$HIGH                          = 00000000
$$LIMIT                         = 00000001
$$LOW                           = FFFFFFFF
$$MNSW                          = 00000001
$$MXSW                          = 00000001
BLD_COMMON                        0000027E R      03
BLD_DLCKFND                       0000038A R      03
BLD_REDO_SRCH                     00000268 R      03
BLD_SRCHDLCK                      00000278 R      03
BLD_TIMESTAMP_RQST                00000270 R      03
BUG$_LOCKMGRERR                   ******** X      03
CA$_MEASURE                     = 00000002
CDRP$K_CM_LONG_LENGTH           = 00000064
CDRP$L_MSGBLD                   = 0000004C
CDRP$L_VAL1                     = 0000002C
CDRP$L_VAL10                    = 00000060
CDRP$L_VAL3                     = 00000034
CDRP$L_VAL5                     = 0000003C
CDRP$L_VAL7                     = 00000044
CDRP$L_VAL9                     = 0000005C
CDRP$W_CDRPSIZE                 = 00000008
CHECK_TIMESTAMP                   00000156 R      03
CLSMSG$B_FACILITY               = 00000008
CLSMSG$B_FUNC                   = 00000009
CLSMSG$K_FAC_LCK                = 00000002
CLU$GL_CLUB                       ******** X      03
CLUB$L_LOCAL_CSID               = 00000060
CLUB$W_MEMSEQ                   = 000000AC
CNX$DEALL_MSG_BUF_CSB             ******** X      03
CNX$INIT_CDRP                     ******** X      03
CNX$RCV_REJECT                    ******** X      03
CNX$SEND_MSG                      ******** X      03
DEALL_DLCK_MSG                    0000049F R      03
DYN$C_FRK                       = 00000008
EXE$ALONONPAGED                   ******** X      03
EXE$DEANONPAGED                   ******** X      03
EXE$FORK_WAIT                     ******** X      03
EXE$GL_ABSTIM                     ******** X      03
EXE$GL_INTSTKLM                   ******** X      03
EXE$GQ_SYSTIME                    ******** X      03
FKB$K_LENGTH                    = 00000018
GET_TIMESTAMP                     000000AF R      03
IPL$_SYNCH                      = 00000008
LCK$ALLOC_LONGCDRP                000004AD R      03
LCK$BREAK_DEADLOCK                ******** X      03
LCK$CVT_ID_TO_LKB                 00000454 RG     03
LCK$DISPATCH                      ******** X      03
LCK$DLCKEXIT                      ******** X      03
LCK$GL_EXTRASTK                   ******** X      03
LCK$GL_IDTBL                      ******** X      03
LCK$GL_MAXID                      ******** X      03
LCK$GL_PRCMAP                     ******** X      03
LCK$GL_TIMOUTQ                    ******** X      03
LCK$GL_TS_CSID                    ******** X      03
LCK$GL_WAITTIME                   ******** X      03
LCK$GQ_BITMAP_EXP                 ******** X      03
LCK$RCV_DLCKFND                   000003A0 RG     03
LCK$RCV_REDO_SRCH                 000003F0 RG     03
LCK$RCV_SRCHDLCK                  0000028E RG     03
LCK$RCV_TIMESTAMP_RQST            00000101 RG     03
LCK$SND_DLCKFND                   00000346 RG     03
LCK$SND_REDO_SRCH                 000003BB RG     03
LCK$SND_SRCHDLCK                  000001A0 RG     03
LCK$SND_TIMESTAMP_RQST            00000000 RG     03
LCK$SRCH_RESDLCK                  ******** X      03
LCK$V_NODLCKWT                  = 00000009
LKB$B_STATE                     = 00000036
LKB$B_TSLT                      = 0000004E
LKB$K_CONVERT                   = 00000000
LKB$K_GRANTED                   = 00000001
LKB$K_WAITING                   = FFFFFFFF
LKB$L_ASTQFL                    = 00000000
LKB$L_CSID                      = 00000058
LKB$L_DLCKPRI                   = 00000024
LKB$L_DUETIME                   = 00000018
LKB$L_LKID                      = 00000030
LKB$L_OWNQBL                    = 00000044
LKB$L_OWNQFL                    = 00000040
LKB$L_PID                       = 0000000C
LKB$L_REMLKID                   = 00000054
LKB$L_RSB                       = 00000050
LKB$M_TIMOUTQ                   = 00000040
LKB$V_MSTCPY                    = 00000004
LKB$V_TIMOUTQ                   = 00000006
LKB$W_FLAGS                     = 00000028
LKB$W_STATUS                    = 0000002A
LKMSG$B_TSLT                    = 0000000E
LKMSG$K_DLCKFND                 = 0000000B
LKMSG$K_DLM_LENGTH              = 00000034
LKMSG$K_REDO_SRCH               = 0000000C
LKMSG$K_SRCHDLCK                = 0000000A
LKMSG$K_TSRQST                  = 00000009
LKMSG$L_NEXTLKID                = 00000030
LKMSG$L_ORIGCSID                = 00000018
LKMSG$L_ORIGEPID                = 00000010
LKMSG$L_ORIGLKID                = 00000014
LKMSG$L_VCTMCSID                = 0000002C
LKMSG$L_VCTMLKID                = 00000028
LKMSG$L_VCTMPRI                 = 00000024
LKMSG$Q_BITMAP_EXP              = 0000001C
LKMSG$W_MEMSEQ                  = 0000000C
LKMSG_BFR                         00000018 R      02
LKMSG_FKB                         00000000 R      02
LOCKFRAME                       = 00000018
MAX_TSLT                        = 00000005
PCB$L_DLCKPRI                   = 0000010C
PCB$L_LOCKQFL                   = 00000104
PMS$GL_DLCKMSGS_IN                ******** X      03
PMS$GL_DLCKMSGS_OUT               ******** X      03
RCV_DLCK_MSG                      0000048C R      03
REDO_SRCH                         000003F8 R      03
```

```
RSB$L_CSID                      = 00000038
SCH$GL_PCBVEC                     ******** X   03
SEND_DLCK_MSG                     0000047A R   03
TSLT_UNITS                      = 0007A120
WAIT_FOR_POOL                     000004CB R   03
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$$$040	0000004C (76.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
$$$020	0000051C (1308.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.03	00:00:01.40
Command processing	136	00:00:00.41	00:00:01.55
Pass 1	451	00:00:11.62	00:00:44.05
Symbol table sort	0	00:00:01.79	00:00:05.56
Pass 2	248	00:00:02.90	00:00:15.85
Symbol table output	15	00:00:00.09	00:00:00.56
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	889	00:00:16.86	00:01:08.99

```
The working set limit was 2100 pages.
99692 bytes (195 pages) of virtual memory were used to buffer the intermediate code.
There were 100 pages of symbol table space allocated to hold 1684 non-local and 57 local symbols.
1432 source lines were read in Pass 1, producing 20 object records in Pass 2.
28 pages of virtual memory were used to define 26 macros.
```

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYSLOA.OBJ]CLUSTER.MLB;1	1
_255DUA28:[SYS.OBJ]LIB.MLB;1	13
_255DUA28:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	20

```
1790 GETS were required to define 20 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:DSTRDLCK/OBJ=OBJ$:DSTRDLCK MSRC$:DSTRDLCK/UPDATE=(ENH$:DSTRDLCK)+EXECML$/LIB+LIB$:CLUSTER/LIB
```

CSPOPCOM
LIS

CSPWAIT
LIS

CSPRCPCAC
LIS

CSPCJFRES
LIS

CSPQUORUM
LIS

DISTRLKI
LIS

CSPMOUNT
LIS

CSPVECTOR
LIS

CSPCLIENT
LIS

DSTRDLCK
LIS

DSTRLOCK
LIS